Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»
Филиал
«Минский радиотехнический колледж»

Учебный предмет
«Конструирование программ и языки программирования»

Инструкция
по выполнению лабораторной работы №32
«Модификация, вставка и удаление записей в наборе данных»

Минск 2024 г.

Лабораторная работа № 32

Тема работы: «Модификация, вставка и удаление записей в наборе данных»

1 Цель работы

Сформировать умения и навыки модификации, вставки и удаление записей в наборе данных

2 Задание

Реализовать основные операции к БД из лабораторной работы №28 (модификация, вставка и удаление записей), поиск и сортировку данных.

3 Оснащение работы

ПК, среда Visual Studio 2019, MSword.

4 Основные теоретические сведения

Основные операции с БД:

1. Выбор данных (SELECT):
   - Используется для извлечения данных из таблицы.
   - Пример: SELECT * FROM таблица WHERE условие;
2. Вставка данных (INSERT):
   - Добавляет новые записи в таблицу.
   - Пример: INSERT INTO таблица (столбец1, столбец2) VALUES (значение1, значение2);
3. Обновление данных (UPDATE):
   - Изменяет существующие записи в таблице.
   - Пример: UPDATE таблица SET столбец1 = новое_значение WHERE условие;
4. Удаление данных (DELETE):
   - Удаляет записи из таблицы.
   - Пример: DELETE FROM таблица WHERE условие;

Поле varbinary в базе данных предназначено для хранения переменного количества бинарных данных. Оно может быть полезно, например, для хранения изображений, видео, или других бинарных файлов.

Для разработки приложения с возможностью реализации основных операций с БД и работой с бинарными полями создадим форму, представленную на рисунке 1.

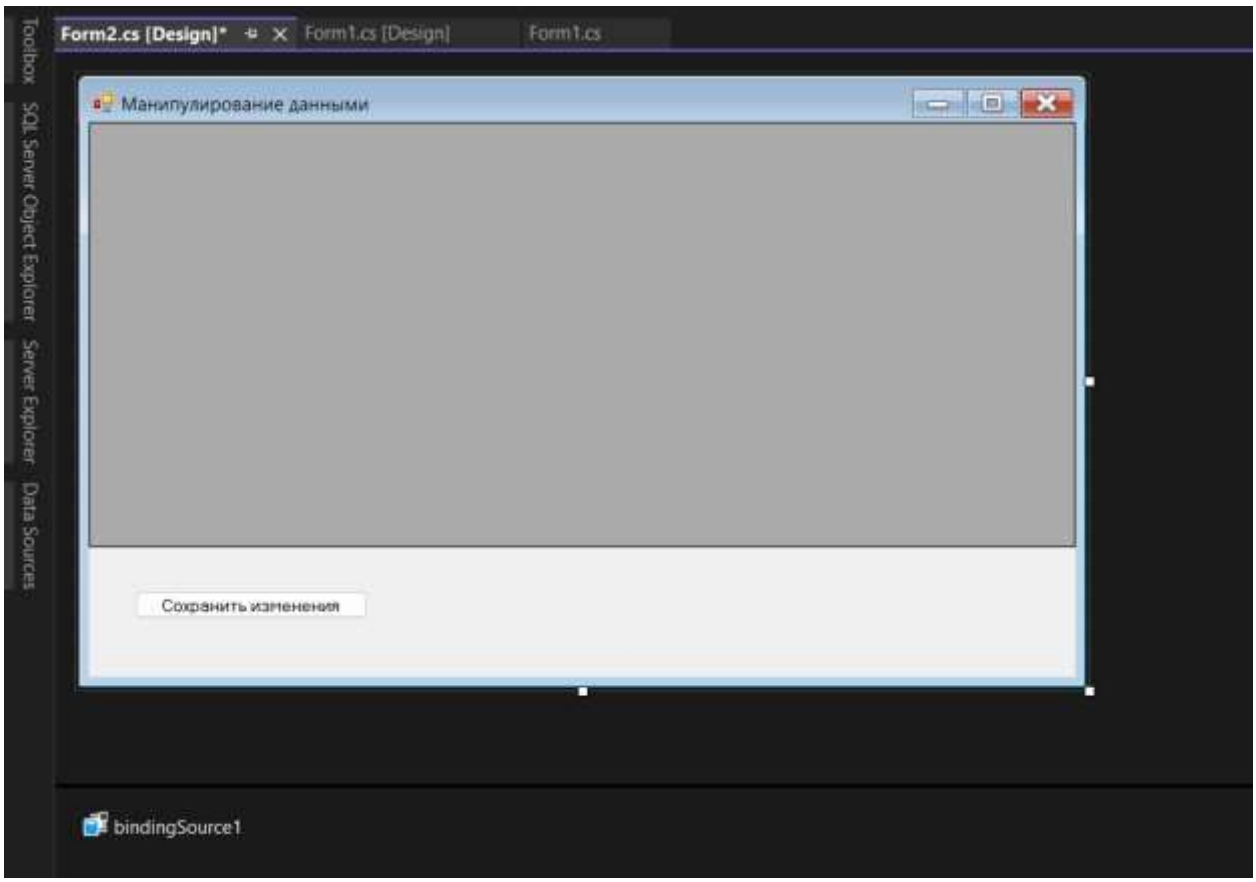

Рисунок 1 – Форма для работы с данными из БД

На данной форме изначально располагаются компоненты DataGridView, bindingSource и Button.

Сформируем строку подключения к базе данных «Кофейня» из предыдущей лабораторной работы.

Для этого выберем компонент bindingSource1, располагающийся на форме, и в панели свойств выберем свойство DataSource. Далее нам необходимо выбрать пункт Добавить новый источник данных в проект (Add Project Data Source), представленный на рисунке 2.

Рисунок 2 – Добавление нового источника данных

В открывшемся окне выбираем пункт База данных (Database) и нажимаем Далее (Next) (рисунок 3).

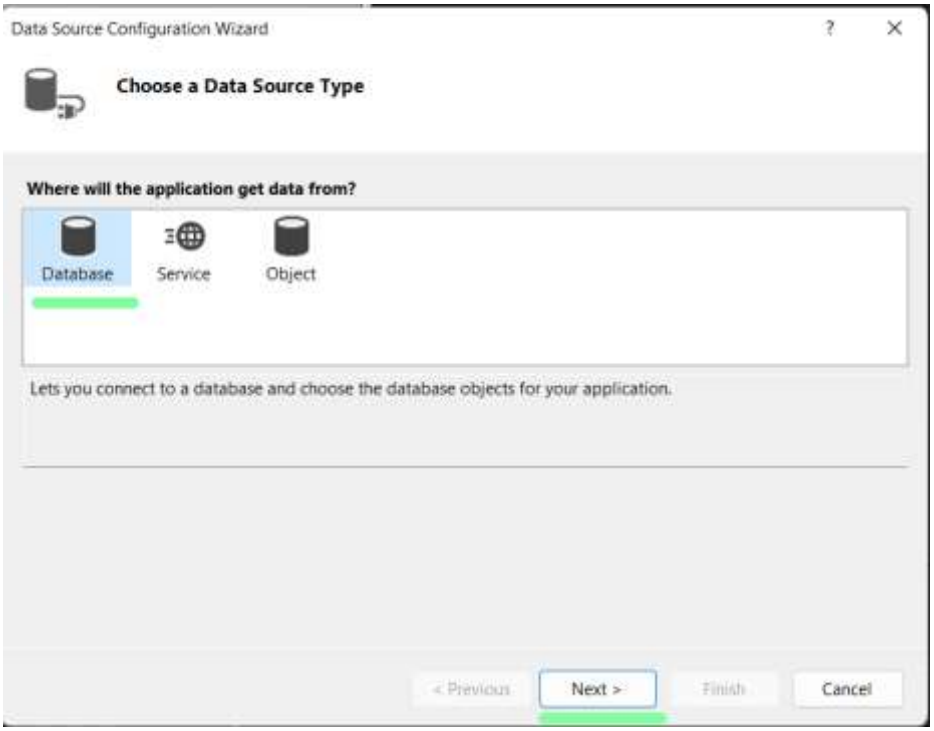

Рисунок 3 – Менеджер настройки источника данных

Далее оставляем все без изменений (рисунок 4).

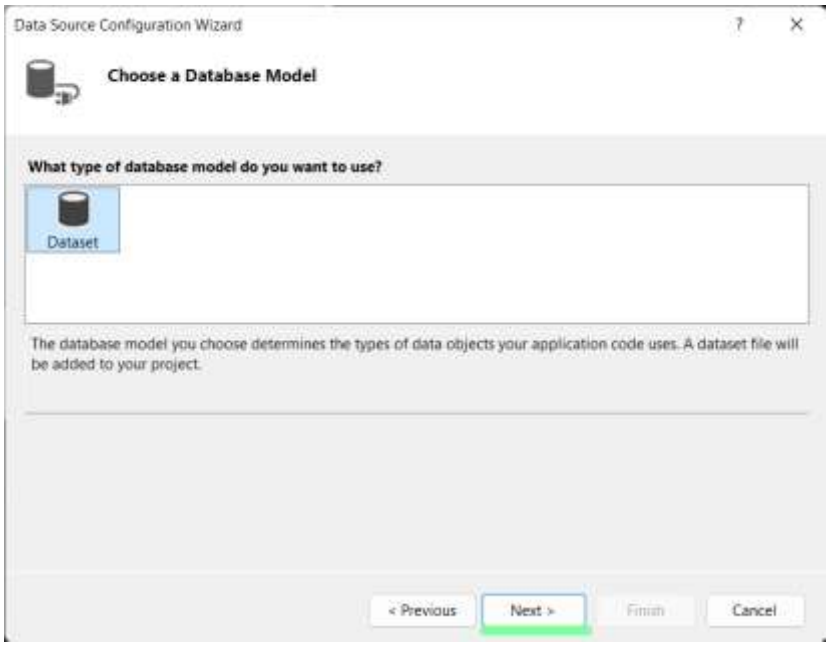

Рисунок 4 – Выбор модели БД

Далее необходимо выбрать строку подключения (рисунок 5).

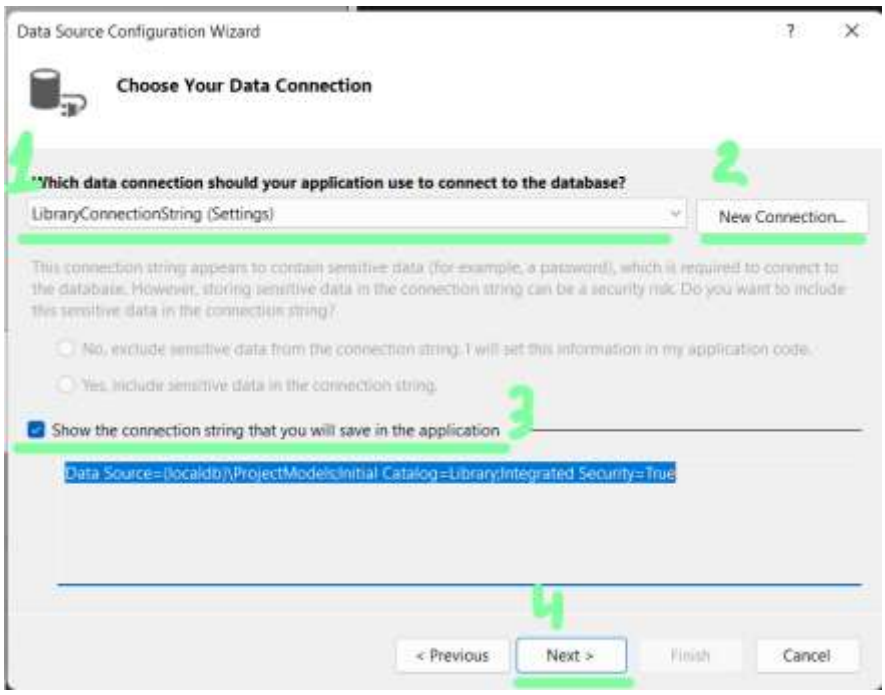

Рисунок 5 – Выбор строки подключения

Если строки подключения нет в проекте или Вы хотите использовать другую строку подключения, необходимо нажать на кнопку Новое подключение (New Connection). Вам откроется окно, представленное на рисунке 6.

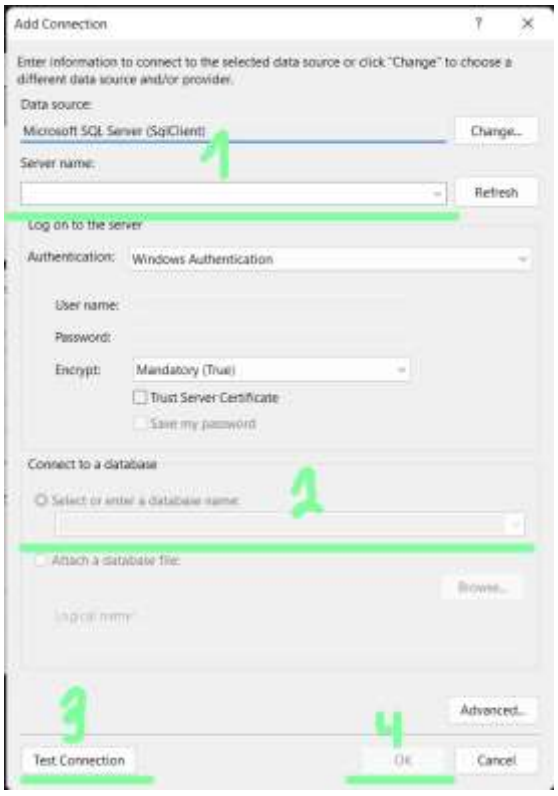

Рисунок 6 – Добавление нового подключения к БД

Первое, что следует заполнить/выбрать – имя сервера. Затем, можно переходить к выбору БД. Имена БД автоматически добавляются при вводе имени сервера. Пример заполнения полей представлен на рисунке 7.

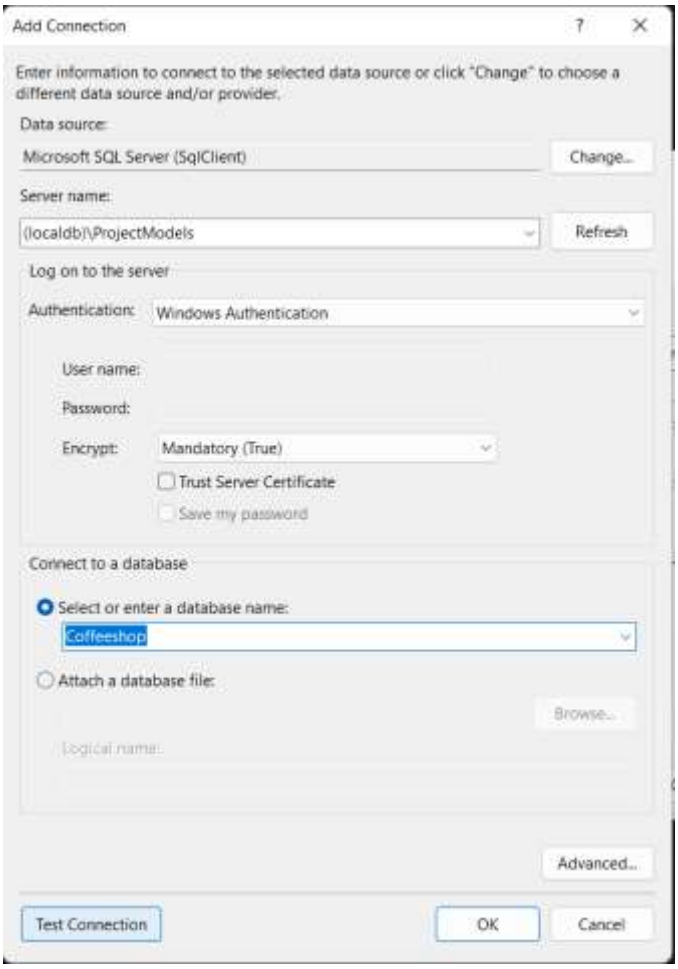

Рисунок 7 – Выбор сервера и БД

После заполнения всех полей, необходимо проверить подключение нажав соответствующую кнопку. Если все хорошо, Вы увидите сообщение, представленное на рисунке 8.

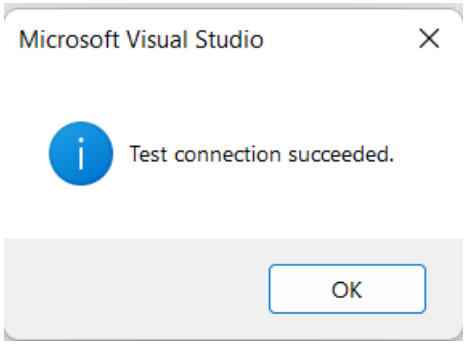

Рисунок 8 – Сообщение об успешном подключении к БД

Далее нажимаем Ок и возвращаемся к окну с формированием строки подключения. Выбираем чекбокс, предлагающий показать строку подключения, которая будет храниться в проекте и нажимаем Далее (Next).

В следующем окне ничего не меняем и нажимаем Далее (Next).

Далее нам предложат выбрать таблицы БД, которые мы хотим использовать. Можно выбрать сразу все, одну или несколько. Для примера выберем все таблицы (рисунок 9).

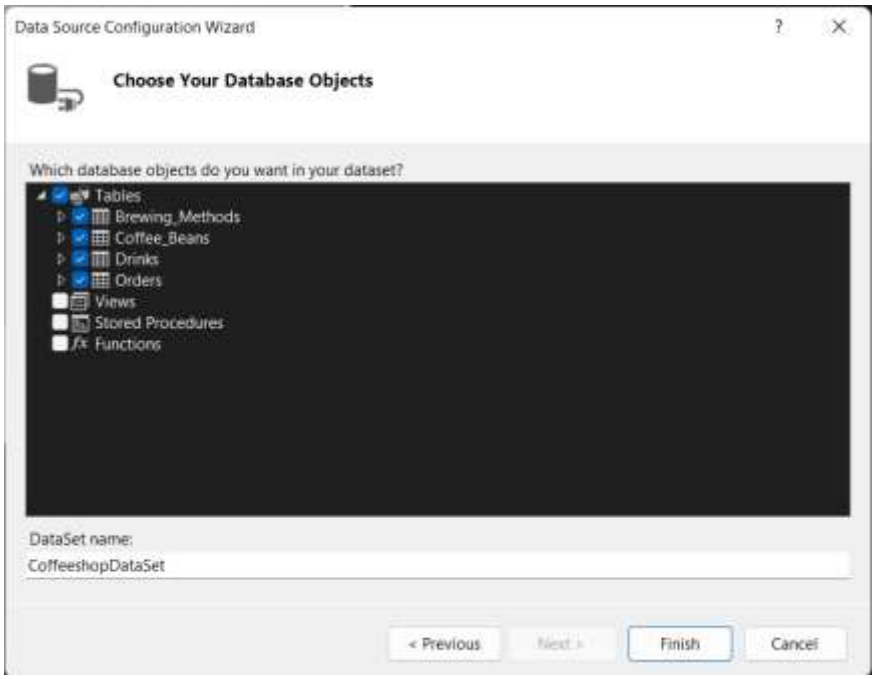

Рисунок 9 – Выбор объектов БД

Далее нажимаем кнопку Завершить (Finish).

В свойстве DataMember компонента bindingSource выбираем необходимую для работы таблицу БД (рисунок 10).

Рисунок 10 – Выбор таблицы БД

Свяжем bindingSource и dataGridView. Для этого нажмем на dataGridView с формы. Рядом с компонентом появится небольшая стрелка. При нажатии на нее открывается меню с выбором источника данных (рисунок 11). Выберем bindingSource в качестве источника данных.

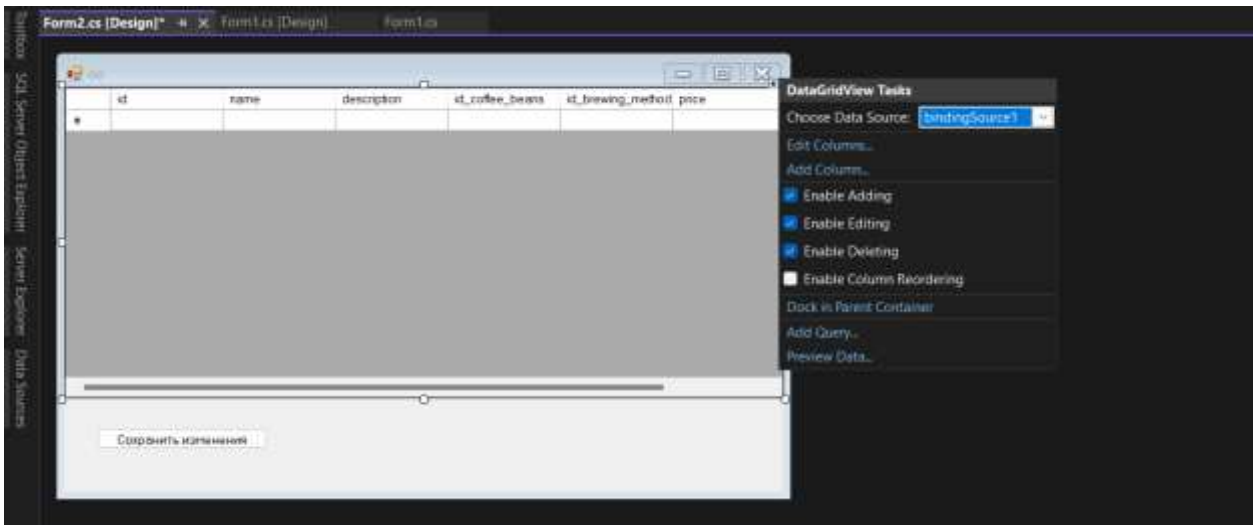

Рисунок 11 – Выбор источника данных для dataGridView

После этого, можем запустить проект и проверить, выводятся ли данные из таблицы (рисунок 12).

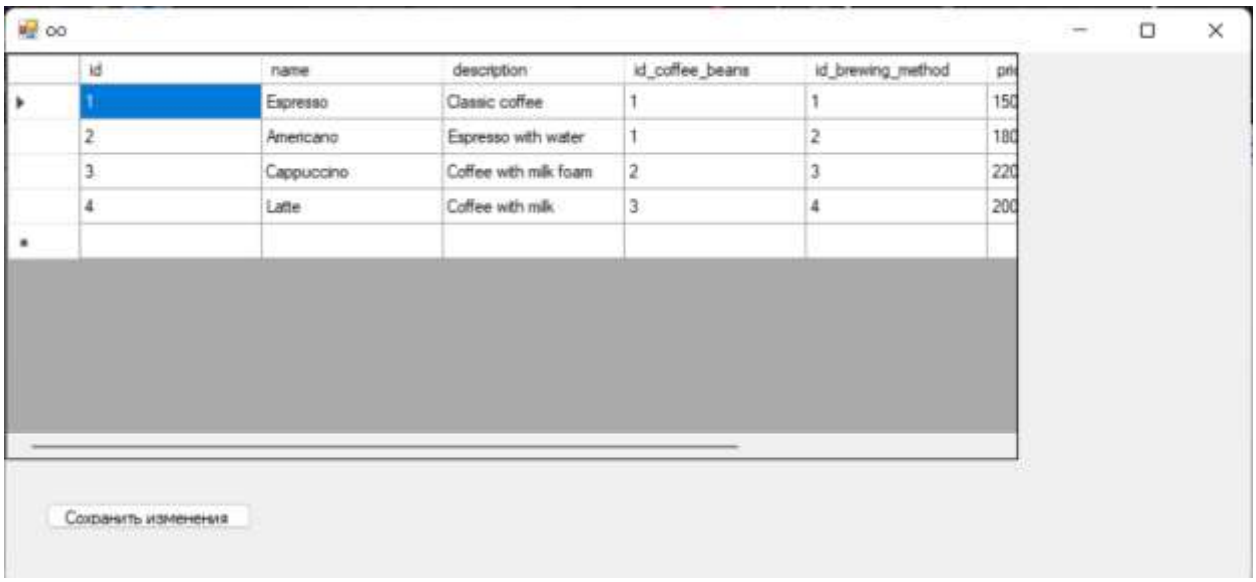

Рисунок 12 – Отображение данных из таблицы БД на форме

Далее можно настроить отображение столбцов таблицы, размеры dataGridView и т.д.

Для выполнения основных операций над БД можно пользоваться различными средствами – чистыми SQL-запросами, средства C#, фреймворки и т.д.

Для примера воспользуемся средствами C#.

В средства для работы с БД входят компоненты DataSet и TableAdapter.

Данные компоненты появляются автоматически при добавлении соединения с БД и выводом данных в dataGridView.

DataSet и TableAdapter являются частями технологии ADO.NET в .NET Framework (или его аналогах). Эти компоненты предоставляют набор средств для работы с данными в приложениях, позволяя удобно получать, изменять, и взаимодействовать с данными из баз данных.

DataSet представляет собой в памяти набор данных, который может содержать несколько таблиц данных, взаимосвязанных отношений, а также схему и другую информацию о данных.

Основные концепции, связанные с DataSet:

— таблицы (Tables) – DataSet может содержать несколько таблиц данных, каждая из которых представляет отдельный набор данных;

— отношения (Relations) – DataSet может также содержать отношения между таблицами, что упрощает работу с взаимосвязанными данными;

— схема данных (Schema) – DataSet хранит информацию о структуре данных в виде схемы, что включает в себя информацию о таблицах, их столбцах, отношениях и т.д.;

— события и изменения данных – DataSet предоставляет события, позволяющие отслеживать изменения данных, такие как добавление, удаление или изменение строк в таблицах.

TableAdapter служит в качестве посредника между DataSet и источником данных, обеспечивая доступ к данным базы данных и управление операциями CRUD (Create, Read, Update, Delete).

Основные возможности и особенности TableAdapter:

— запросы к базе данных – TableAdapter включает в себя SQL-запросы или хранимые процедуры для выполнения операций выборки, вставки, обновления и удаления данных в базе данных;

— автоматическая генерация команд – инструменты дизайнера позволяют автоматически генерировать команды (SQL) для обновления данных в соответствии с изменениями в DataSet;

— интеграция с DataSet – 98шгTableAdapter связывается с конкретной таблицей в DataSet, обеспечивая согласованность данных между базой данных и DataSet;

— кеширование данных – TableAdapter может выполнять кеширование данных для оптимизации доступа и уменьшения нагрузки на базу данных;

— поддержка транзакций – TableAdapter может работать в контексте транзакций, что позволяет выполнять несколько операций как единое целое.

Для дальнейшей работы будем пользоваться таблицами БД, представленными на рисунке 13.

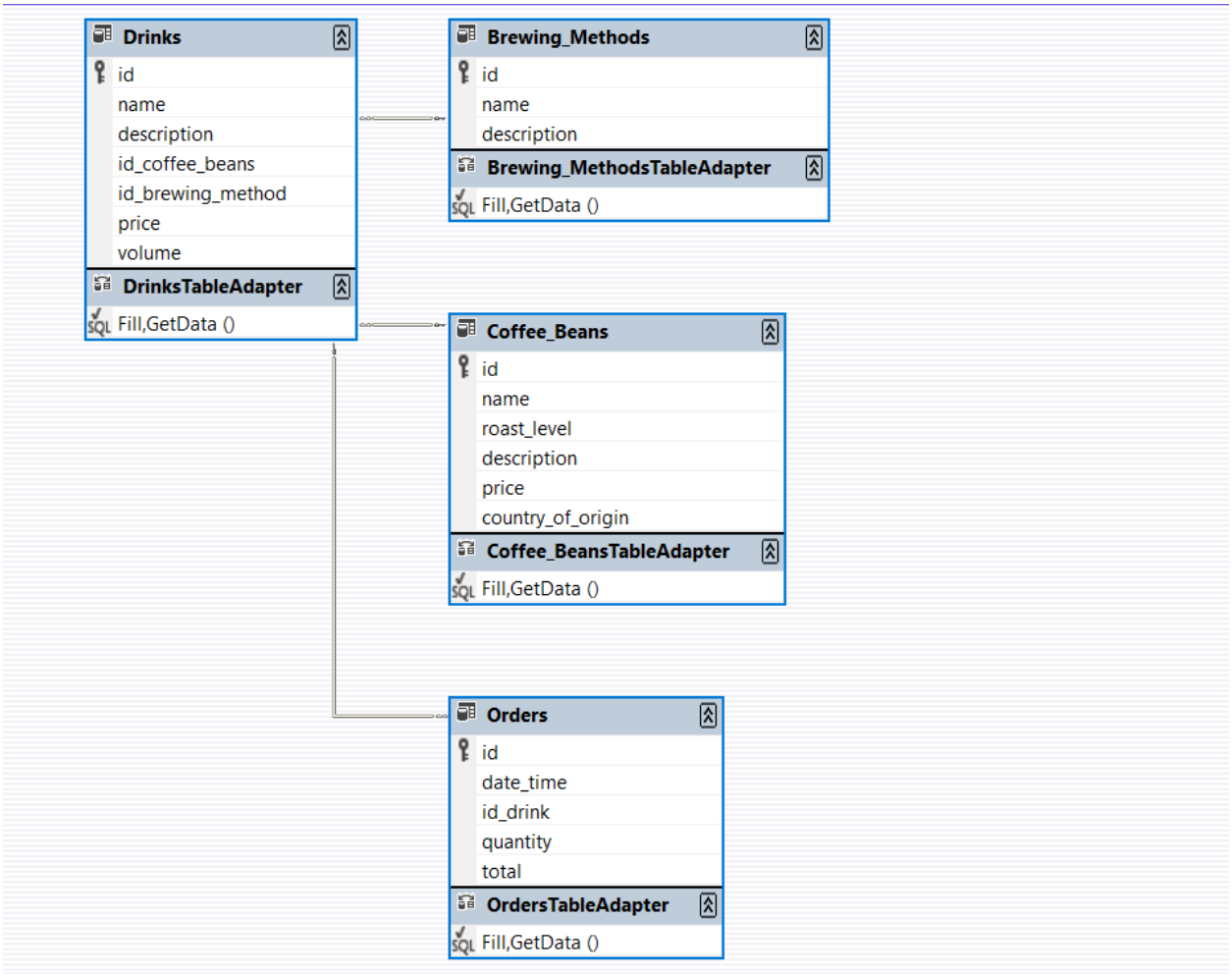

Рисунок 13 – Схема БД

Чтобы получить доступ к схеме БД, можно воспользоваться контекстным меню для DataSet (рисунок 14).

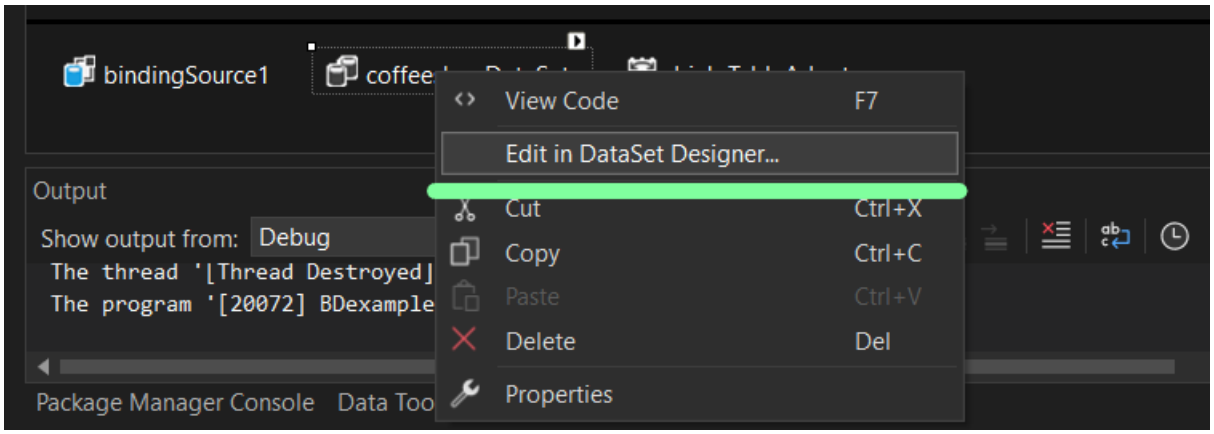

Рисунок 14 – Получение доступа к схеме БД

Воспользуемся TableAdapter для реализации CRUD нашей таблицы в БД.

Редактировать, удалять, изменять и просматривать данные мы будем непосредственно в dataGridView, к которому привязана таблица БД. Сохранение изменений будет происходить по щелчку мыши на кнопку.

Для того, чтобы сохранить изменения, напишем код для кнопки Сохранить изменения.

Для этого, щелкнем два раза по кнопке. Откроется код C# с уже зарегистрированным событием нажатия кнопки (рисунок 15).

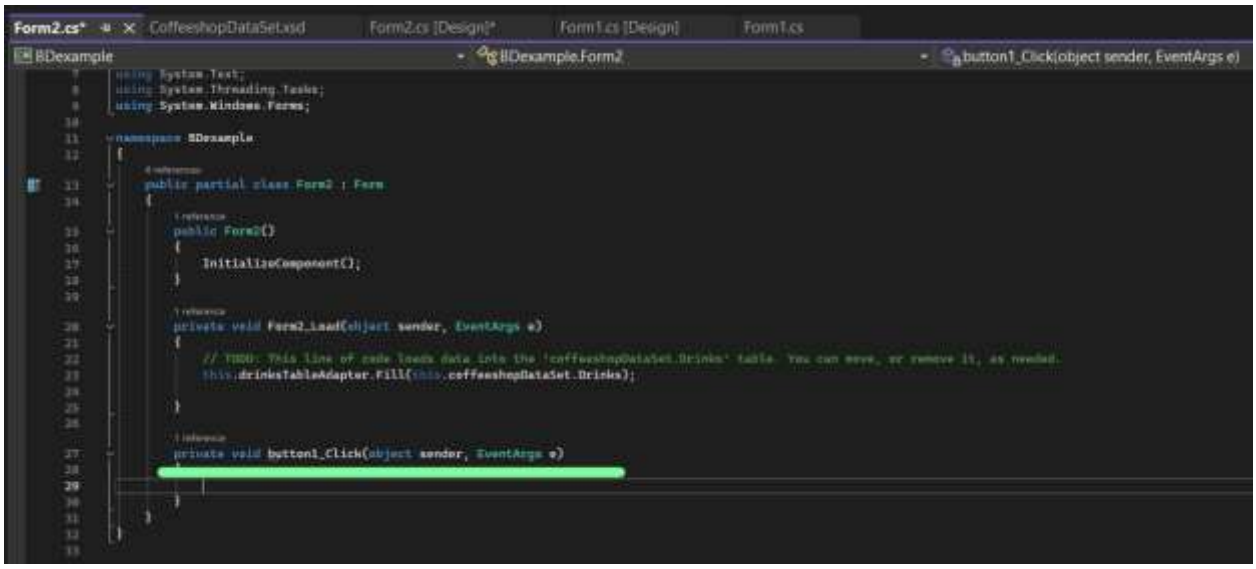

Рисунок 15 – Шаблон события для кнопки

В кнопку напишем код, представленный рисунке 16.

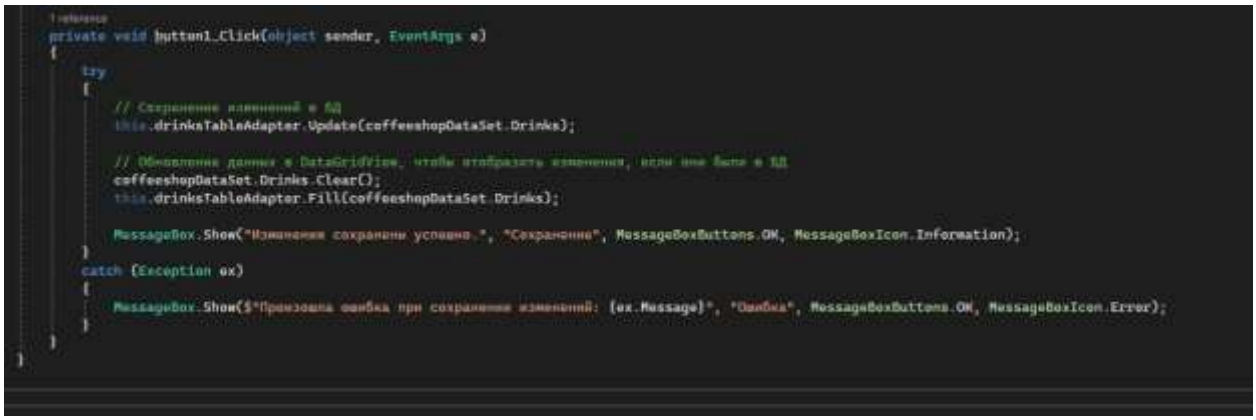

Рисунок 16 – Код для сохранения изменений

Теперь попробуем изменить стоимость напитка с id=4. Для этого просто отредактируем данные в таблице и нажмем на кнопку для сохранения изменений. После повторного открытия приложения изменения сохраняются. В примере выросла стоимость латте с 200 рублей до 220 рублей. При сохранении данных получаем сообщение, что изменения сохранены успешно (рисунок 17).

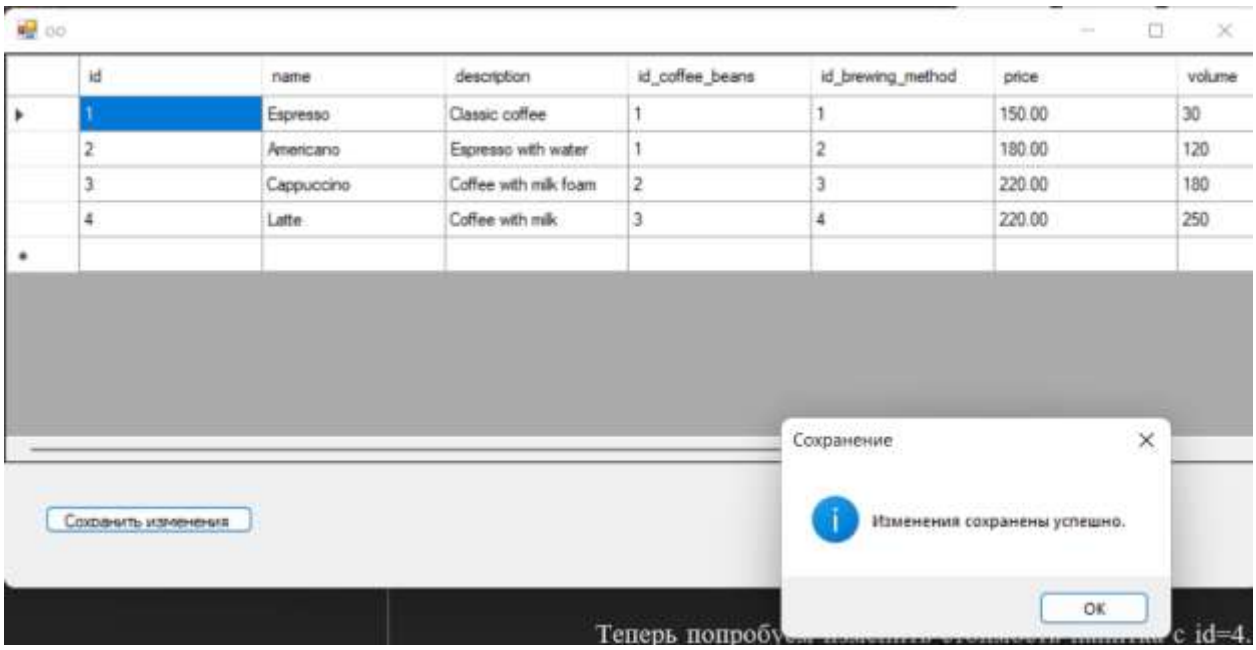

Рисунок 17 – Сохранение изменений в БД

Кроме выполнения операций по CRUD мы можем реализовать поддержку таких операций, как поиск и сортировка данных.

Сортировка данных может происходить автоматически путем нажатия на название столбцов таблицы на этапе выполнения программы (рисунок 18).

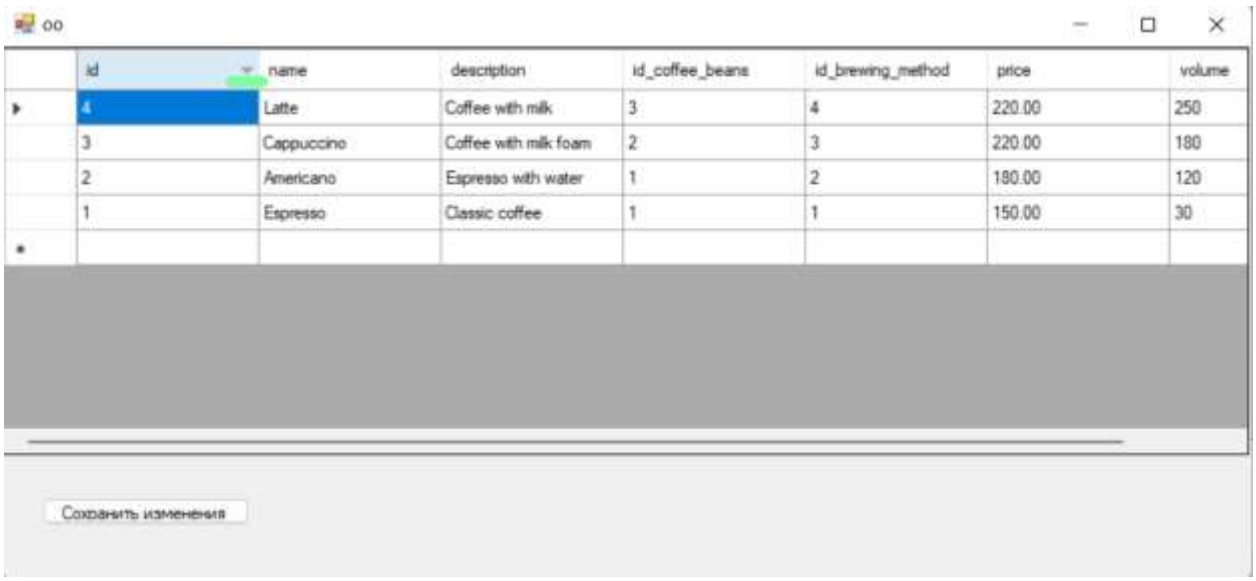

Рисунок 18 – Встроенная сортировка данных через dataGridView

Для реализации поиска по данным из таблицы добавим на форму поле для ввода (TextBox), подпись (Label) и кнопку (Button) (рисунок 19).

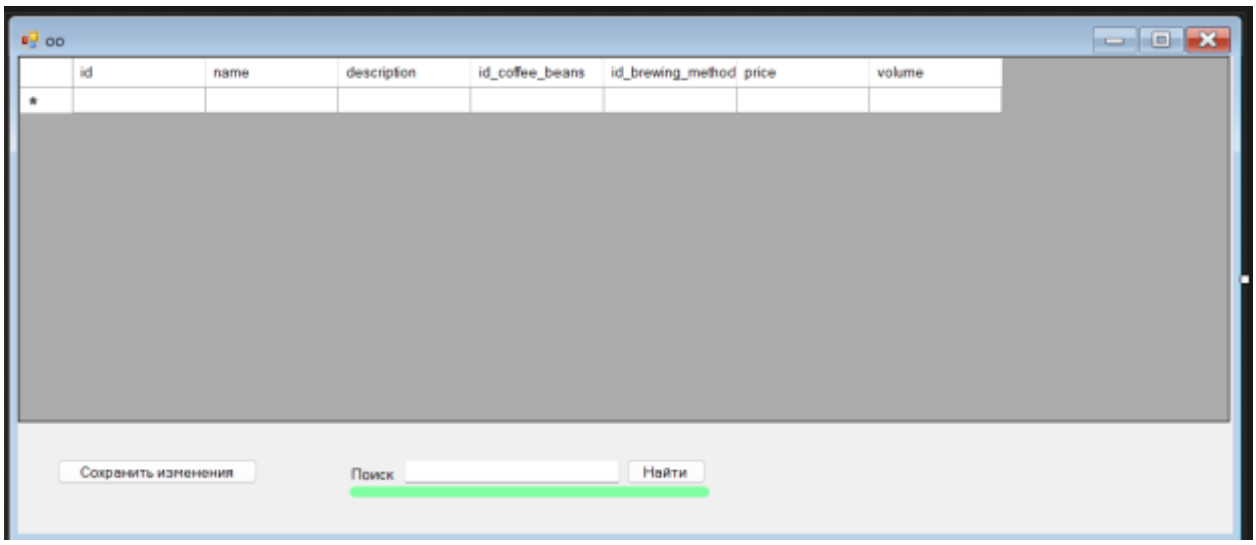

Рисунок 19 – Добавление элементов для осуществления поиска

Запрограммируем кнопку Найти на поиск данных в таблице.

Для этого зарегистрируем событие по нажатию кнопки и напишем код, представленный на рисунке 20.

```
private void button2_Click(object sender, EventArgs e)
{
    string searchQuery = textBox1.Text.Trim();

    // Очистим DataGridView перед новым поиском
    coffeeshopDataSet.Clear();

    // Используем SQL-запрос для поиска в таблице Drinks
    string sqlQuery = "SELECT * FROM Drinks WHERE name LIKE @search OR description LIKE @search";

    using (SqlConnection connection = new SqlConnection("Data Source=(localdb)\\ProjectModels;Initial Catalog=CoffeeShop;Integrated Security=True"))
    {
        connection.Open();

        using (SqlDataAdapter adapter = new SqlDataAdapter(sqlQuery, connection))
        {
            adapter.SelectCommand.Parameters.AddWithValue("@search", "%" + searchQuery + "%");

            adapter.Fill(coffeeshopDataSet, "Drinks");
        }
    }

    // Обновим DataGridView
    dataGridView1.DataSource = coffeeshopDataSet.Tables["Drinks"];
}
```

Рисунок 20 – Реализация функции поиска

В данном коде используется запрос, который работает по двум полям таблицы БД – name и description, позволяя пользователю искать данные либо по названию напитка, либо по описанию. Запрос принимает в себя один параметр – @search, который заполняется тем, что ввел пользователь для поиска.

Далее, с помощью классов SqlConnection и SqlDataAdapter выполняем запрос и переносим результаты в datagridview.

При создании объекта класса SqlConnection в конструктор передается строка подключения к БД. Чтобы ее найти, можно перейти в Обозреватель серверов (Server Explorer) и выбрать подключение к источнику данных, с которым сейчас работаем. В окне свойств в разделе Connection (свойство Connection String) увидим необходимую строку подключения (рисунок 21).

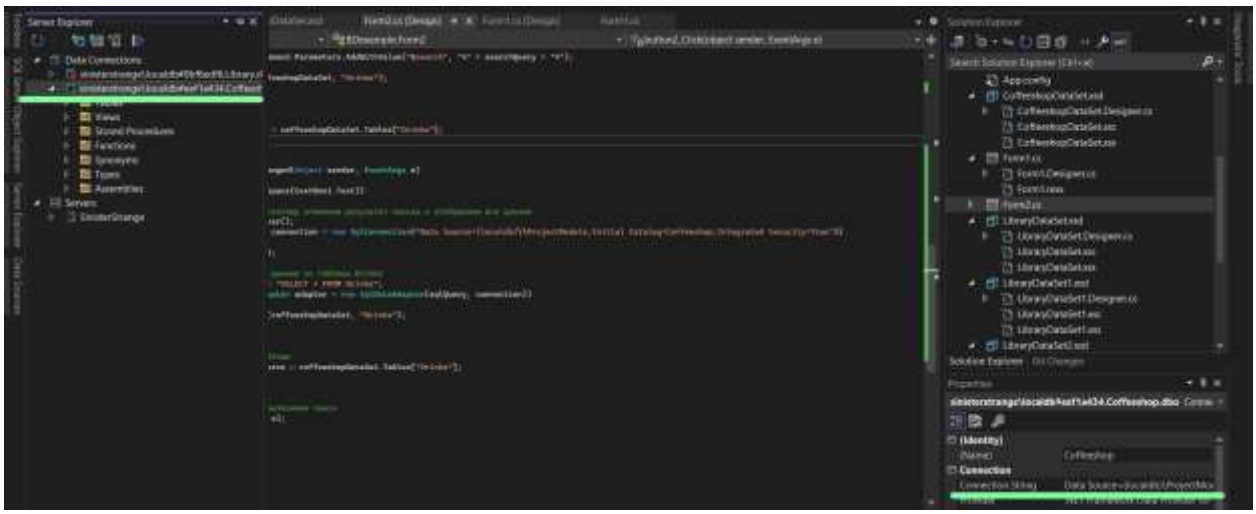

Рисунок 21 – Расположение строки подключения к БД

Для того, чтобы использование классов не приводило к ошибке, в коде необходимо прописать использование System.Data.SqlClient (рисунок 22).

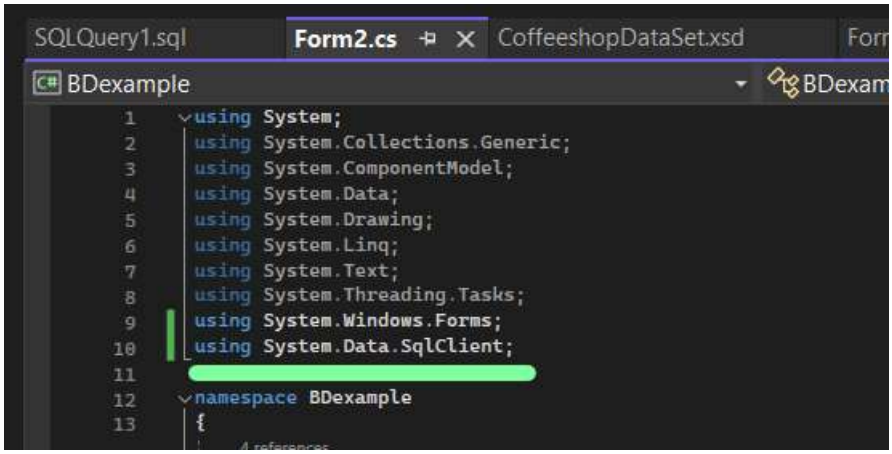

Рисунок 22 – Подключение System.Data.SqlClient

Чтобы не помещать большое количество кнопок на форму, для отмены поиска, можно воспользоваться событием TextChanged у копонента textBox (рисунок 23).

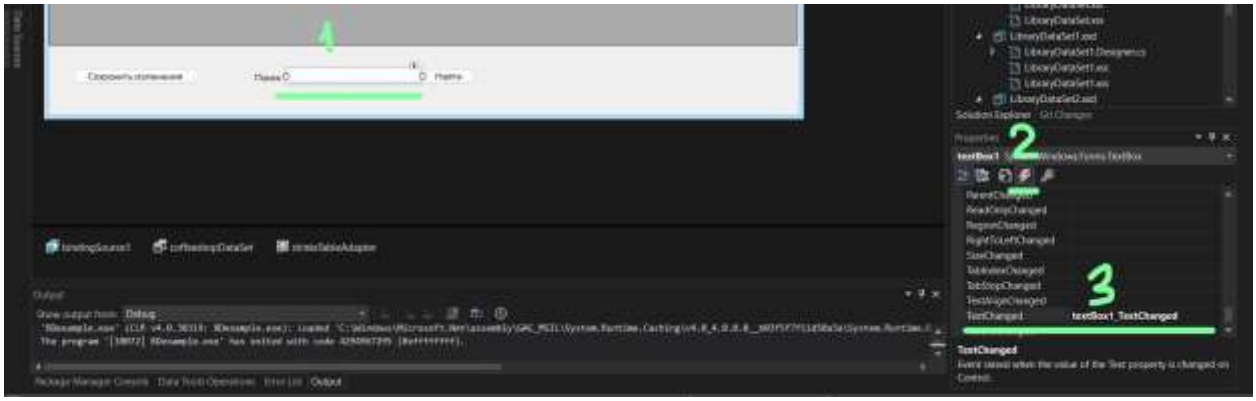

Рисунок 23 – Регистрация события изменения текста у компонента textBox

Для регистрации события выберем компонент textBox, лежащий на форме. Далее перейдем к окну свойств элемента и нажмем на значок молнии. Мы попадаем в окно всех событий, которые могут быть зарегистрированы для элемента. Находим событие TextChanged и нажимаем на поле рядом два раза. Нам открывается код нашей формы с уже зарегистрированным событием. Осталось применить код для отмены результатов поиска (рисунок 24).

```
private void textBox1_TextChanged(object sender, EventArgs e)
{
    if (string.IsNullOrWhiteSpace(textBox1.Text))
    {
        // Очистка TextBox, поэтому отменяем результат поиска и отображаем все данные
        coffeeshopDataSet.Clear();
        using (SqlConnection connection = new SqlConnection("Data Source=(localdb)\\ProjectModels;Initial Catalog=Coffeeshop;Integrated Security=True"))
        {
            connection.Open();

            // Загружаем все данные из таблицы Drinks
            string sqlQuery = "SELECT * FROM Drinks";
            using (SqlDataAdapter adapter = new SqlDataAdapter(sqlQuery, connection))
            {
                adapter.Fill(coffeeshopDataSet, "Drinks");
            }
        }

        // Обновляем DataGridView
        dataGridView1.DataSource = coffeeshopDataSet.Tables["Drinks"];
    }
    else
    {
        // TextBox не пуст, выполняем поиск
        button2_Click(sender, e);
    }
}
```

Рисунок 24 – Код для вывода всех данных из таблицы при очищении textBox

5. Порядок выполнения работы

1. Выделить ключевые моменты задачи.
2. Построить алгоритм и теоретическую объектную модель решения задачи.
3. Запрограммировать полученные алгоритмы и объектную модель.

6. Форма отчета о работе

Лабораторная работа № ___

Номер учебной группы _____

Фамилия, инициалы учащегося _____

Дата выполнения работы _____
Тема работы: _____
Цель работы: _____
Оснащение работы: _____
Результат выполнения работы: _____

7. Контрольные вопросы и задания

1. Перечислите и кратко опишите основные CRUD-операции.
2. Кратко опишите процесс подключения БД к компонентам bindingSource и dataGridView.
3. Что нужно прописать в коде для корректной работы классов SqlConnection и SqlDataAdapter?
4. Кратко опишите процесс регистрации события для компонента на форме.

8. Рекомендуемая литература

1. Рихтер, Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C# / Дж. Рихтер. СПб. : Изд-во Питер, 2021. 896 с.

2. Прайс, М. Дж. C# 10 и .NET 6. Современная кросс-платформенная разработка / М. Дж. Прайс. СПб : Изд-во Питер, 2023. 848 с.

3. Васильев, А.Н. Программирование на C# для начинающих. Особенности языка / А.Н. Васильев. М. : Эксмо, 2022. 528 с.

4. Фримен, А. ASP.NET Core 3 с примерами на C# для профессионалов / А. Фримен. СПб. : Изд-во Вильямс, 2021. 1184 с.